**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ**

**МОСКОВСКИЙ ЭНЕРГЕТИЧЕСКИЙ ИНСТИТУТ**

**Лабораторная работа №3**

**Функции и пользовательские типы данных.**

**Выполнила:**

А-02-13

Головтеева А.Н.

**Принял:**

Козлюк Д.А.

Москва, 2015

1. **В программе из ЛР № 1 выделить код для преобразования температуры между шкалами в функцию double convert(double temperature, char from, char to);**

**При некорректных значениях аргументов печатать из функции предупреждения.**

double convert(double t, char from, char to)

{ if (from == 'C' || from == 'F' || from == 'K')

{

if (from == 'K')

{

if (t < 0)

cout <<"ErrorTempInKelvin \n";

else

{

switch (to)

{

case 'K':

return t;

break;

case 'C':

return (t - 273.15);

break;

case 'F':

return (9/5\*(t - 273.15) + 32);

break;

}

}

}

else if (from == 'C')

{

switch (to)

{

case 'C':

return t;

break;

case 'K':

return (t - 273.15);

break;

case 'F':

return (9/5 \* t + 32);

break;

}

}

else switch (to)

{

case 'F':

return t;

break;

case 'C':

return (5/9 \* (t - 32));

break;

case 'K':

return (5/9 \* (t - 32) + 273.15);

break;

}

}

else

cout <<"unknown scale.\n";}

**2. Завести перечисление Error для кодов ошибок, которые могут возникать в функции convert(), чтобы использовать его в пунктах 3 и 3.**

enum Error {not\_error, , error\_tempreture\_K }

not\_error – нет ошибки;

error\_scale – ошибка при вводе шкалы;

error\_tempretureK – ошибка, связанная с не отрицательностью температуры в Кельвинах.

1. **Реализовать обработку ошибок вовне convert()по кодам возврата.**

#include "sdt.h"

enum Error {not\_error, error\_scale, error\_tempreture\_K };

Error convert(double t, char from, char to, double& result)

{

if (from == 'C' || from == 'F' || from == 'K')

{

if (from == 'K')

{

if (t < 0)

return error\_tempreture\_K;

else

{

switch (to)

{

case 'K':

{

result = t;

return not\_error;

break;

}

case 'C':

result = t - 273.15;

return not\_error;

break;

case 'F':

result = 9/5\*(t - 273.15) + 32;

return not\_error;

break;

}

}

}

else if (from == 'C')

{

switch (to)

{

case 'C':

result = t;

return not\_error;

break;

case 'K':

result = t - 273.15;

return not\_error;

break;

case 'F':

result = 9/5 \* t + 32;

return not\_error;

break;

}

}

else switch (to)

{

case 'F':

result = t;

return not\_error;

break;

case 'C':

result = 5/9 \* (t - 32);

return not\_error;

break;

case 'K':

result = 5/9 \* (t - 32) + 273.15;

return not\_error;

break;

}

}

else

return error\_scale;

}

int main ()

{

vector<double> degsC, degsK, degsF;

char sc;

double t,c,k,f;

while (cin)

{

cout <<"Enter temperature (C,F,K) = ";

cin >> t >> sc;

if (!cin)

break;

else

{

switch (convert(t, sc, 'K', k))

{

case error\_scale:

cerr<< "unknown scale.\n";

break;

case error\_tempreture\_K:

cerr<<"ErrorTempInKelvin. \n";

break;

case not\_error :

degsK.push\_back(k);

convert(t,sc,'C',c);

degsC.push\_back(c);

convert(t,sc,'F',f);

degsF.push\_back(f);

break;

default:

cerr<< "Unknown error!";

}

}

}

cout <<"\n";

cout <<"C \t\t" <<"K \t\t" <<"F \t\t \n";

for (int i = 0; i < degsK.size(); ++i)

{

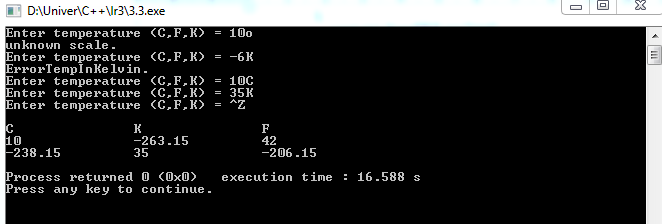
cout << degsC[i] << "\t\t";

cout << degsK[i] << "\t\t";

cout << degsF[i] << "\t\t" << "\n";

}

}



1. **Реализовать обработку ошибок вовне convert()через код последней ошибки.**

#include "sdt.h"

enum Error {not\_error, error\_scale, error\_tempreture\_K };

Error last\_error = not\_error;

Error get\_last\_error()

{

return last\_error;

}

double convert(double t, char from, char to)

{

if (from == 'C' || from == 'F' || from == 'K')

{

if (from == 'K')

{

if (t < 0)

last\_error = error\_tempreture\_K;

else

{

switch (to)

{

case 'K':

{

return(t);

last\_error = not\_error;

break;

}

case 'C':

return(t - 273.15);

last\_error = not\_error;

break;

case 'F':

return(9/5\*(t - 273.15) + 32);

last\_error = not\_error;

break;

}

}

}

else if (from == 'C')

{

switch (to)

{

case 'C':

return(t);

last\_error = not\_error;

break;

case 'K':

return(t - 273.15);

last\_error = not\_error;

break;

case 'F':

return(9/5 \* t + 32);

last\_error = not\_error;

break;

}

}

else switch (to)

{

case 'F':

return(t);

last\_error = not\_error;

break;

case 'C':

return(5/9 \* (t - 32));

last\_error = not\_error;

break;

case 'K':

return(5/9 \* (t - 32) + 273.15);

last\_error = not\_error;

break;

}

}

else

last\_error = error\_scale;

}

int main ()

{

vector<double> degsC, degsK, degsF;

char sc;

double t,c,k,f;

while (cin)

{

cout <<"Enter temperature (C,F,K) = ";

cin >> t >> sc;

if (!cin)

break;

else

{

c = convert(t, sc, 'C');

k = convert(t, sc, 'K');

f = convert(t, sc, 'F');

switch (get\_last\_error())

{

case error\_scale:

cerr<< "unknown scale.\n";

break;

case error\_tempreture\_K:

cerr<<"ErrorTempInKelvin. \n";

break;

case not\_error:

degsC.push\_back(c);

degsK.push\_back(k);

degsF.push\_back(f);

}

}

last\_error = not\_error;

}

cout <<"\n";

cout <<"C \t\t" <<"K \t\t" <<"F \t\t \n";

for (int i = 0; i < degsK.size(); ++i)

{

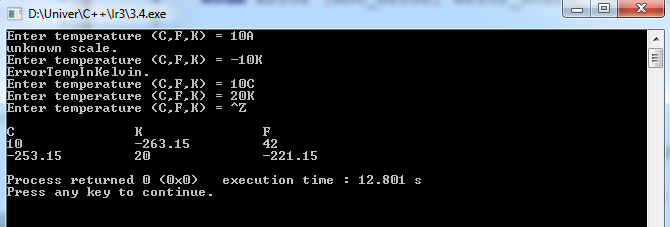
cout << degsC[i] << "\t\t";

cout << degsK[i] << "\t\t";

cout << degsF[i] << "\t\t" << "\n";

}

}



**5. Реализовать обработку ошибок вовне функции через механизм исключительных ситуаций.**

#include "sdt.h"

double convert(double t, char from, char to)

{

if (from == 'C' || from == 'F' || from == 'K')

{

if (from == 'K')

{

if (t < 0)

throw invalid\_argument("ErrorTempInKelvin.\n");

else

{

switch (to)

{

case 'K':

return(t);

break;

case 'C':

return(t - 273.15);

break;

case 'F':

return(9/5\*(t - 273.15) + 32);

break;

}

}

}

else if (from == 'C')

{

switch (to)

{

case 'C':

return(t);

break;

case 'K':

return(t - 273.15);

break;

case 'F':

return(9/5 \* t + 32);

break;

}

}

else switch (to)

{

case 'F':

return(t);

break;

case 'C':

return(5/9 \* (t - 32));

break;

case 'K':

return(5/9 \* (t - 32) + 273.15);

break;

}

}

else

{ throw logic\_error("Unknown scale!"); }

throw exception();

}

int main ()

{

vector<double> degsC, degsK, degsF;

char sc;

double t;

while (cin)

{

cout <<"Enter temperature (C,F,K) = ";

cin >> t >> sc;

if (!cin)

break;

else

{

try

{

degsC.push\_back(convert(t,sc,'C'));

degsK.push\_back(convert(t,sc,'K'));

degsF.push\_back(convert(t,sc,'F'));

}

catch (const invalid\_argument& e)

{

cerr << e.what() << "\n";

}

catch (const logic\_error& le)

{

cerr << le.what() << "\n";

}

catch (...)

{

cerr << "Unknown error!\n";

}

}

}

cout <<"\n";

cout <<"C \t\t" <<"K \t\t" <<"F \t\t \n";

for ( int i = 0; i < degsC.size(); ++i)

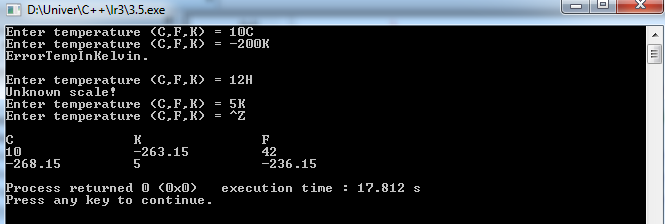
{

cout << degsC[i] << "\t\t";

cout << degsK[i] << "\t\t";

cout << degsF[i] << "\t\t" << "\n";

}}

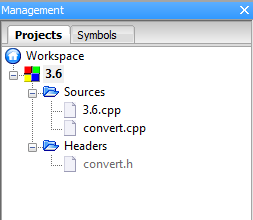


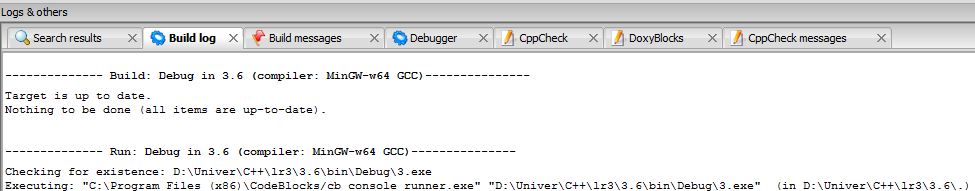
**6. Создать (изменить) проект так, чтобы в нем были:**

1) основной файл исходного кода с функцией main();

2) заголовочный файл с объявлением функции convert();

3) файл реализации с определением функции convert().





**7. Добавить в заголовочный файл определение перечисления Scale, заменить тип параметров to и from функции convert() на Scale.**

Содержание заголовочного файла convert.h:

enum Scale

{

Kelvin,

Celsius,

Fahrenheit

};

double convert(double t, Scale from, Scale to);

Scale change(char scale);

Содержание файла реализации convert.сpp:

#include "sdt.h"

#include "convert.h"

double convert(double t, Scale from, Scale to)

{

if (from == Celsius || from == Kelvin || from == Fahrenheit)

{

if (from == Kelvin)

{

if (t < 0)

throw invalid\_argument("ErrorTempInKelvin.\n");

else

{

switch (to)

{

case Kelvin:

{

return(t);

break;

}

case Celsius:

return(t - 273.15);

break;

case Fahrenheit:

return(9/5\*(t - 273.15) + 32);

break;

}

}

}

else if (from == Celsius)

{

switch (to)

{

case Celsius:

return(t);

break;

case Kelvin:

return(t - 273.15);

break;

case Fahrenheit:

return(9/5 \* t + 32);

break;

}

}

else switch (to)

{

case Fahrenheit:

return(t);

break;

case Celsius:

return(5/9 \* (t - 32));

break;

case Kelvin:

return(5/9 \* (t - 32) + 273.15);

break;

}

}

else

{

throw logic\_error("Unknown scale!");

}

throw exception();

}

Scale change(char sc)

{

switch (sc)

{

case 'C':

return Celsius;

break;

case 'F':

return Fahrenheit;

break;

case 'K':

return Kelvin;

break;

default:

throw invalid\_argument("Unknown scale.\n");

}

}

**8. Добавить возбуждение исключения в функции convert() в том случае, если алгоритма преобразования между заданными шкалами в ней не реализовано.**

Алгоритм преобразования между шкалами уже реализован и возбуждение исключения присутствует -invalid\_argument("Unknown scale.\n"). Этим решается проблема, возникающая при некорректном вводешкал.

**9. Добавить в заголовочный файл определение структуры Temperature, представляющей температуру со шкалой, с конструктором**

**Temperature(double value, char scale).**

Описание структуры:

struct Temperature

{

Temperature(double value,char scale);

double value;

char scale;

};

Реализация конструктора:

Temperature::Temperature(double value, char scale)

{

this->value = value;

this->scale = scale;

}

**10. Модернизировать основную программу: использовать Temperature вместо отдельных переменных для значения и шкалы, а также для хранения в векторе.**

**Основная программа:**

#include "sdt.h"

#include "convert.h"

int main ()

{

Temperature temp{0,'C'};

vector<Temperature> degsC, degsK, degsF;

char sc;

double c,k,f,s;

while (cin)

{

cout <<"Enter temperature (C,F,K) = ";

cin >> temp.value >> temp.scale;

if (!cin)

break;

else

{

try

{

degsC.push\_back(convert(temp,Celsius));

degsK.push\_back(convert(temp,Kelvin));

degsF.push\_back(convert(temp,Fahrenheit));

}

catch (const invalid\_argument& e)

{

cerr << e.what() << "\n";

}

catch (const logic\_error& le)

{

cerr << le.what() << "\n";

}

catch (...)

{

cerr << "Unknown error!\n";

}

}

}

cout <<"\n";

cout <<"C \t\t" <<"K \t\t" <<"F \t\t \n";

for ( int i = 0; i < degsC.size(); ++i)

{

cout << degsC[i].value << "\t\t";

cout << degsK[i].value << "\t\t";

cout << degsF[i].value << "\t\t" << "\n";

}

}

**Функция convert так же претерпела изменения:**

Temperature convert(Temperature t, Scale to )

{

Temperature data{0,'C'};

if (t.scale == 'C' || t.scale == 'K' || t.scale == 'F')

{

if (t.scale == 'K')

{

if (t.value < 0)

throw invalid\_argument("ErrorTempInKelvin.\n");

else

{

switch (to)

{

case Kelvin:

{

data.value = t.value;

data.scale =Kelvin;

break;

}

case Celsius:

{

data.value = t.value - 273.15;

data.scale =Celsius;

break;

}

case Fahrenheit:

{

data.value = 9/5\*(t.value - 273.15) + 32;

data.scale =Fahrenheit;

break;

}

}

}

}

else if (t.scale == 'C')

{

switch (to)

{

case Celsius:

{

data.value = t.value;

data.scale =Celsius;

break;

}

case Kelvin:

{

data.value = t.value + 273.15;

data.scale =Kelvin;

break;

}

case Fahrenheit:

{

data.value =(9/5 \* t.value + 32);

data.scale =Fahrenheit;

break;

}

}

}

else switch (to)

{

case Fahrenheit:

{

data.value = t.value;

data.scale =Fahrenheit;

break;

}

case Celsius:

{

data.value = 5/9 \* (t.value - 32);

data.scale =Celsius;

break;

}

case Kelvin:

{

data.value = 5/9 \* (t.value - 32) + 273.15;

data.scale =Kelvin;

break;

}

}

}

else

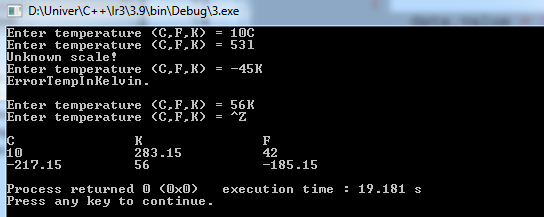
{

throw logic\_error("Unknown scale!");

}

return data;

}



**11. Перегрузить операторы ввода и вывода в поток для перечисления Scale и для структуры Temperature (итого четыре новые функции). Задействовать их в основной программе.**

**Заголовочный файл convert.h:**

istream& operator>> (istream& input, Temperature& temp);

ostream& operator>> (ostream& output, const Temperature& temp);

istream& operator>> (istream& input, Scale& sc);

ostream& operator>> (ostream& output, const Scale& sc);

**Файл реализации tempreture.cpp:**

istream& operator>> (istream& input, Temperature& temp)

{

input >> temp.value >> temp.scale;

return input;

}

ostream& operator<< (ostream& output, const Temperature& temp)

{

output << temp.value << temp.scale;

return output;

}

**Файл реализации сonvert.cpp:**

istream& operator>> (istream& input, Scale& sc)

{

char s;

input>> s;

switch (s)

{

case 'C':

sc = Celsius;

break;

case 'K':

sc = Kelvin;

break;

case 'F':

sc = Fahrenheit;

break;

default :

input.setstate(ios\_base::failbit);

}

return input;

}

ostream& operator<< (ostream& output, const Scale& sc)

{

switch (sc)

{

case Celsius:

output << 'C';

break;

case Kelvin:

output << 'K';

break;

case Fahrenheit:

output << 'F';

}

return output;

}

**12. Перегрузить оператор сравнения для двух температур (<), сложения двух температур (+), деления температуры на число (/) и соответствующие им операторы сокращенного присваивания (+=, /=). Задействовать их для того, чтобы по окончании ввода данных печатать наибольшую, наименьшую и среднюю температуру.**

**Перегрузка операторов в convert.h:**

bool operator>(const Temperature lhs, const Temperature rhs);

Temperature operator+ (const Temperature& lhs, const Temperature& rhs);

Temperature operator/ (Temperature& lhs, const int del);

Temperature operator+= (Temperature& lhs, const Temperature& rhs);

Temperature operator/= (Temperature& lhs, const int del);

**В файл реализации tempreture.cpp добавили:**

bool operator>(const Temperature& lhs, const Temperature& rhs)

{

Temperature T = convert(rhs, lhs.scale);

return lhs.value > T.value;

}

Temperature operator+(const Temperature& lhs, const Temperature& rhs)

{

Temperature T = convert(rhs, lhs.scale);

return {lhs.value + T.value,

lhs.scale};

}

Temperature operator/(Temperature& lhs, const int del)

{

return {lhs.value / del,

lhs.scale};

}

Temperature operator+=(Temperature& lhs, const Temperature& rhs)

{

Temperature T = convert(rhs, lhs.scale);

lhs.value += T.value;

return lhs;

}

Temperature operator/=(Temperature& lhs, const int del)

{

lhs.value /= del;

return lhs;

}

**13. Выделить вычисление наибольшей, наименьшей и средней температур в функцию. Типы и способы передачи параметров выбрать самостоятельно**.

**сonvert.h:**

Temperature maxtemp(vector<Temperature> temp);

Temperature mintemp(vector<Temperature> temp);

Temperature mean(vector<Temperature> temp);

**tempreture.cpp:**

Temperature maxtemp(vector<Temperature> temp)

{

Temperature maxt = temp[0];

for (int i = 1; i < temp.size(); i++)

{

if (temp[i].value > maxt.value)

maxt = temp[i];

}

return maxt;

}

Temperature mintemp(vector<Temperature> temp)

{

Temperature mintemp {0, Celsius};

for (int i = 0; i < temp.size(); i++)

{

if (temp[i].value < mintemp.value)

mintemp = temp[i];

}

return mintemp;

}

Temperature mean(vector<Temperature> temp)

{

Temperature mean {0, Celsius};

for (int i=0; i < temp.size(); i++)

{

mean += temp[i];

}

mean /= temp.size();

return mean;

}

**Основная программа:**

#include "sdt.h"

#include "convert.h"

int main ()

{

Temperature input {0,Kelvin};

vector<Temperature> degsC, degsK, degsF;

while (cin)

{

cout <<"Enter temperature (C,F,K) = ";

if (!(cin>> input))

break;

else

{

try

{

degsC.push\_back(convert(input,Celsius));

degsK.push\_back(convert(input,Kelvin));

degsF.push\_back(convert(input,Fahrenheit));

}

catch (const invalid\_argument& e)

{

cerr << e.what() << "\n";

}

catch (const logic\_error& le)

{

cerr << le.what() << "\n";

}

catch (...)

{

cerr << "Unknown error!\n";

}

}

}

cout <<"\n";

cout <<"C \t\t" <<"K \t\t" <<"F \t\t \n";

for ( int i = 0; i < degsC.size(); ++i)

{

cout << degsC[i].value << "\t\t";

cout << degsK[i].value << "\t\t";

cout << degsF[i].value << "\t\t" << endl;

}

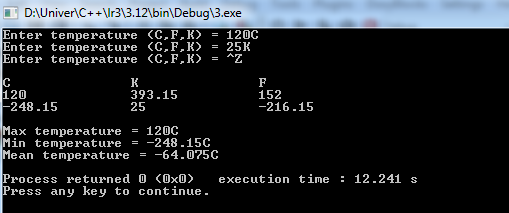
cout<<endl;

cout <<"Max temperature = " << maxtemp(degsC) << endl;;

cout <<"Min temperature = " << mintemp(degsC) << endl;

cout <<"Mean temperature = " << mean(degsC) << endl;

}



**14. Добавить сохранение всех введенных температур в текстовый файл перед окончанием работы программы.**

**Изменение в основной программе:**

ofstream output("lab3.txt", ios\_base::trunc);

cout <<"\n";

output<<"C \t\t" <<"K \t\t" <<"F \t\t \n";

cout <<"C \t\t" <<"K \t\t" <<"F \t\t \n";

for ( int i = 0; i < degsC.size(); ++i)

{

cout << degsC[i].value << "\t\t";

cout << degsK[i].value << "\t\t";

cout << degsF[i].value << "\t\t" << endl;

output << degsC[i].value << "\t\t";

output << degsK[i].value << "\t\t";

output << degsF[i].value << "\t\t\n" ;

}

cout<<endl;

output <<"Max temperature = " << maxtemp(degsC) <<endl;

output <<"Min temperature = " << mintemp(degsC) << endl;

output <<"Mean temperature = " << mean(degsC) << endl;

cout <<"Max temperature = " << maxtemp(degsC) << endl;

cout <<"Min temperature = " << mintemp(degsC) << endl;

cout <<"Mean temperature = " << mean(degsC) << endl;

}

